**移动端reset**

body,h1,h2,h3,h4,h5,h6,hr,p,blockquote,dl,dt,dd,ul,ol,li,pre,figcaption,figure,fieldset,legend,button,input,textarea,th,td{margin:0;padding:0}

body,button,input,select,textarea{font:12px/1 Lucida Grande,'Microsoft YaHei',"Lucida Grande","Lucida Sans Unicode",Helvetica,Arial,Verdana,sans-serif}

h1{font-size:18px;font-weight:normal}

h2{font-size:16px;font-weight:normal}

h3{font-size:14px;font-weight:normal}

h4,h5,h6{font-size:100%;font-weight:normal}

address,cite,dfn,em,var{font-style:normal}

code,kbd,pre,samp,tt{font-family:"Courier New",Courier,monospace}

small{font-size:12px}

ul,ol,li{list-style:none}

a{text-decoration:none}

abbr[title],acronym[title]{border-bottom:1px dotted;cursor:help}

q:before,q:after{content:''}

legend{color:#000}

fieldset,img{border:0}

table{border-collapse:collapse;border-spacing:0}

hr{border:0;height:1px}

\*{-ms-word-break:break-all;word-break:break-all;-ms-word-wrap:break-word;word-wrap:break-word;-webkit-tap-highlight-color:rgba(0,0,0,0)}

**给需要的属性加厂家前缀:prefix**

 "-moz-transition": transition,

 "-moz-transform": transform,

 "-ms-transition": transition,

 "-ms-transform": transform,

 "-webkit-transition": transition,

 "-webkit-transform": transform,

 "transition": transition,

 "transform": transform

**四则运算**

**语法：**

**calc()** = calc(四则运算)

**说明：**

**用于动态计算长度值。**

* 需要注意的是，运算符前后都需要保留一个空格，例如：width: calc(100% - 10px)；
* 任何长度值都可以使用calc()函数进行计算；
* calc()函数支持 "+", "-", "\*", "/" 运算；
* calc()函数使用标准的数学运算优先级规则；

**兼容性：**

* 浅绿 = 支持
* 红色 = 不支持
* 粉色 = 部分支持

| **IE** | **Firefox** | **Chrome** | **Safari** | **Opera** | **iOS Safari** | **Android Browser** | **Android Chrome** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6.0-8.0 | 2.0-3.6 | 4.0-18.0 | 3.1-5.1 | 15.0+ | 3.2-5.1 | 2.1-4.3 | 18.0 |
| 9.0+ | 4.0-15.0 -moz- | 19.0-25.0 -webkit- | 6.0 -webkit- | 6.0-6.1 -webkit- | 4.4-4.4.4 | 19.0-25.0 -webkit- |
| 16.0+ | 26.0+ | 6.1+ | 7.0+ | 26.0+ |

**示例：**

<!DOCTYPE html>

<html lang="zh-cmn-Hans">

<head>

<meta charset="utf-8" />

<title>calc()函数\_CSS参考手册\_web前端开发参考手册系列</title>

<meta name="author" content="Joy Du(飘零雾雨), [dooyoe@gmail.com](mailto:dooyoe@gmail.com), [www.doyoe.com](http://www.doyoe.com/)" />

<style>

.test {

    width: calc(100% - 100px);

    background: #eee;

}

</style>

</head>

<body>

<div class="test">我的宽度为100% - 50px</div>

</body>

</html>

相关链接：

<http://www.css88.com/book/css/values/functional/calc%28%29.htm>

<http://www.w3cplus.com/css3/how-to-use-css3-calc-function.html>

calc()对大家来说，或许很陌生，不太会相信calc()是css中的部分。因为看其外表像个函数，既然是函数为何又出现在CSS中呢？这一点 也让我百思不得其解，今天有一同事告诉我，说CSS3中有一个属性能实现自适应的布局，首先让我想到的是box-sizing，但跟我说还可以计算，这让 我不得不想起calc()。因为早先在官网和一些blog上看到相关的介绍，但一直没有深入，也没有自己去测试过。今天花了一下午的时间彻底学习了一下 calc()。于是就有了这篇blog，希望对大家有所帮助。

平时在制作页面的时候，总会碰到有的元素是100%的宽度。众所周知，如果元素宽度为100%时，其自身不带其他盒模型属性设置还好，要是有别的， 那将导致盒子撑破。比如说，有一个边框，或者说有margin和padding，这些都会让你的盒子撑破。我们换句话来说，如果你的元素宽度是100% 时，只要你在元素中添加了border,padding,margin任何一值，都将会把元素盒子撑破（标准模式下，除IE怪异模式）。这样一来就会相当 的麻烦，平时我们碰到这样的现象时，也是相当的谨慎，有时甚至无法解决，只能通过改变结构来实现。就算你通过繁琐的方法实现了，但有于浏览器的兼容性而导 致最终效果不一致。虽然前面介绍的CSS3属性中的box-sizing在一定程度上解决这样的问题，其实今天的calc()函数功能实现上面的效果来得 更简单。

**什么是calc()?**

学习calc()之前，我们有必要先知道calc()是什么？只有知道了他是个什么东东？在实际运用中更好的使用他。

calc() 从字面我们可以把他理解为一个函数function。其实calc是英文单词calculate(计算)的缩写，是css3的一个新增的功能，用来指定元 素的长度。比如说，你可以使用calc()给元素的border、margin、pading、font-size和width等属性设置动态值。为何说 是动态值呢?因为我们使用的表达式来得到的值。不过calc()最大的好处就是用在流体布局上，可以通过calc()计算得到元素的宽度。

**calc()能做什么？**

calc()能让你给元素的做计算，你可以给一个div元素，使用百分比、em、px和rem单位值计算出其宽度或者高度，比如说“width:calc(50% + 2em)”，这样一来你就不用考虑元素DIV的宽度值到底是多少，而把这个烦人的任务交由浏览器去计算。

**calc()语法**

calc()语法非常简单，就像我们小时候学加 （+）、减（-）、乘（\*）、除（/）一样，使用数学表达式来表示：

.elm {

width: calc(expression);

}

其中"expression"是一个表达式，用来计算长度的表达式。

**calc()的运算规则**

calc()使用通用的数学运算规则，但是也提供更智能的功能：

1. 使用“+”、“-”、“\*” 和 “/”四则运算；
2. 可以使用百分比、px、em、rem等单位；
3. 可以混合使用各种单位进行计算；
4. 表达式中有“+”和“-”时，其前后必须要有空格，如"widht: calc(12%+5em)"这种没有空格的写法是错误的；
5. 表达式中有“\*”和“/”时，其前后可以没有空格，但建议留有空格。

**浏览器的兼容性**

浏 览器对calc()的兼容性还算不错，在IE9+、FF4.0+、Chrome19+、Safari6+都得到较好支持，同样需要在其前面加上各浏览器厂 商的识别符，不过可惜的是，移动端的浏览器还没仅有“firefox for android 14.0”支持，其他的全军覆没。

大家在实际使用时，同样需要添加浏览器的前缀

.elm {

/\*Firefox\*/

-moz-calc(expression);

/\*chrome safari\*/

-webkit-calc(expression);

/\*Standard \*/

calc();

}

通过上面的了解，大家对calc()不在那么陌生，但对于实际的运用可能还是不太了解，那么大家就接下来跟我一起动手，通过实例来了解他吧。首先我们来看一个最常用的实例：

<div class="demo">

<div class="box"></div>

</div>

上面的结构很简单，就是一个div.demo的元素中包含了一个div.box的元素，接下来我们一步一步来看其中的变化。

**第一步：添加普通样式：**

.demo {

width: 300px;

background: #60f;

}

.box {

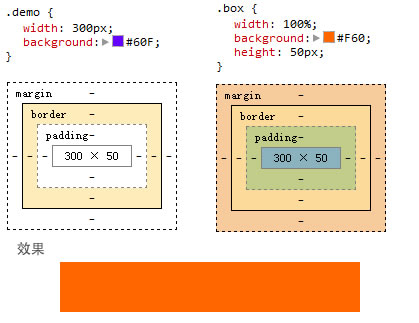
width: 100%;

background: #f60;

height: 50px;

}

此时的效果很简单，就是div.box完全遮盖了div.demo，如下图所示：

[](http://jsfiddle.net/w3cplus/69648/)

**第二步，在div.box上添加border和padding**

这一步很棘手的事情来了，在div.box上添加10px的内距padding，同时添加5px的border：

.demo {

width: 300px;

background: #60f;

}

.box {

width: 100%;

background: #f60;

height: 50px;

**padding: 10px;**

**border: 5px solid green;**

}

为了更好的说明问题，我在div.demo上添加了一个padding：3px 0;

.demo {

width: 300px;

background: #60f;

**padding: 3px 0;**

}

.box {

width: 100%;

background: #f60;

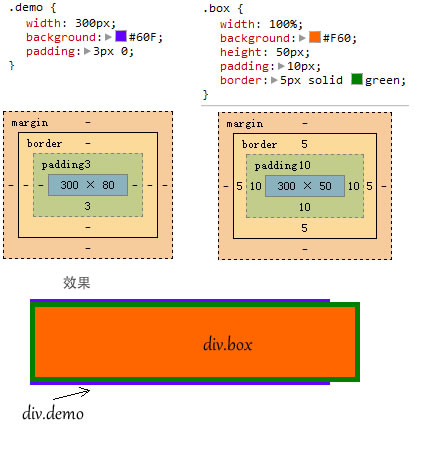
height: 50px;

padding: 10px;

border: 5px solid green;

}

这个时候大家不知道能否想到问题会发生在哪？其实很简单，这个时候div.box的宽度大于了其容器div.demo的总宽度，从而撑破容器伸出来了，如图所示：

[](http://jsfiddle.net/w3cplus/KBUTk/)

**第三步，calc()的运用**

为 了解决撑破容器的问题，以前我们只能去计算div.box的宽度，用容器宽度减去padding和border的值，但有时候，我们苦于不知道元素的总宽 度，比如说是自适应的布局，只知道一个百分值，但其他的值又是px之类的值，这就是难点，死卡住了。随着CSS3的出现，其中利用box-sizing来 改变元素的盒模型类型实使实现效果，但今天我们学习的calc()方法更是方便。

知道总宽度是100%，在这个基础上减去boder的宽度 （5px \* 2 = 10px）,在减去padding的宽度（10px \* 2 = 20px），即"100% - (10px + 5px) \* 2 = 30px" ，最终得到的值就是div.box的width值：

.demo {

width: 300px;

background: #60f;

padding: 3px 0;

}

.box {

background: #f60;

height: 50px;

padding: 10px;

border: 5px solid green;

**width: 90%;/\*写给不支持calc()的浏览器\*/**

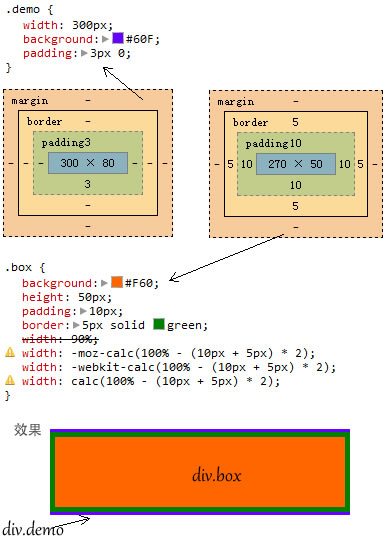
**width:-moz-calc(100% - (10px + 5px) \* 2);**

**width:-webkit-calc(100% - (10px + 5px) \* 2);**

**width: calc(100% - (10px + 5px) \* 2);**

}

这样一来，通过calc()计算后，div.box不在会超出其容器div.demo的宽度，如图所示：

[](http://jsfiddle.net/w3cplus/GeEvb/)

上面是一个简单的实例，接下来我们一起来看一个自适应布局的例子：

上面的demo是一个非常简单而常见的布局效果，在这个布局中，我采用了自适应布局。整个布局包含了“头部”、“主内容”、“边栏”和“脚部”，并且“主内容”居左，“边栏”靠右，具体结构请看DEMO中的html部分。

接下来，我们主要看看css部分：

**1、在body中设置一个内距，并附上一些基本的样式，大家可以根据自己需要进行不同的设置，此例代码如下：**

body {

background: #E8EADD;

color: #3C323A;

padding: 20px;

}

**2、设置主容器“wrapper”的样式**

主容器的宽度是“100% - 20px \* 2”,并且水平居中：

.wrapper {

width: 1024px; /\* Fallback for browsers that don't support the calc() function \*/

width: -moz-calc(100% - 40px);

width: -webkit-calc(100% - 40px);

width: calc(100% - 40px);

margin: auto;

}

给不支持calc()的浏览器设置了一个固定宽度值“1024px”。

**3、给header和footer设置样式**

这个例子中的header和footer很简单，给他们添加了一个内距为20px，其他就是一些基本的样式设置，那么其对应的宽度应该是"100% - 20px \* 2"：

#header {

background: #f60;

padding: 20px;

width: 984px;/\*Fallback for browsers that don't support the calc() function\*/

width: -moz-calc(100% - 40px);

width: -webkit-calc(100% - 40px);

width: calc(100% - 40px);

}

#footer {

clear:both;

background: #000;

padding: 20px;

color: #fff;

width: 984px;/\* Fallback for browsers that don't support the calc() function \*/

width: -moz-calc(100% - 40px);

width: -webkit-calc(100% - 40px);

width: calc(100% - 40px);

}

**4、给主内容设置样式**

给主内容设置了一个8px的边框，20px的内距，并且向左浮动，同时设置了一个向右的外边距“20”px，关键之处，我们主内容占容器宽度的75%，这样一来，主内容的宽度应该是“75% - 8px \* 2 - 20px \* 2”:

#main {

border: 8px solid #B8C172;

float: left;

margin-bottom: 20px;

margin-right: 20px;

padding: 20px;

width: 704px; /\* Fallback for browsers that don't support the calc() function \*/

width: -moz-calc(75% - 20px \* 2 - 8px \* 2);

width: -webkit-calc(75% - 20px \* 2 - 8px \* 2);

width: calc(75% - 20px \* 2 - 8px \* 2);

}

**4、设置右边栏样式**

给边栏设置了一个25%的宽度，其除了包含8px的边框，10px的内距外，还有主内容外距20px也要去掉，不然整个宽度与容器会相差20px,换句话说就会撑破容器掉下来。因此边栏的实际宽度应该是"25% - 10px \* 2 - 8px \* 2 -20px":

#accessory {

border: 8px solid #B8C172;

float: right;

padding: 10px;

width: 208px; /\* Fallback for browsers that don't support the calc() function \*/

width: -moz-calc(25% - 10px \* 2 - 8px \* 2 - 20px);

width: -webkit-calc(25% - 10px \* 2 - 8px \* 2 - 20px);

width: calc(25% - 10px \* 2 - 8px \* 2 - 20px);

}

这样一来，大家就看到了上面demo展现的布局效果。经过此例的学习，大家是不是会觉得使用calc()用于自适应布局是超爽的呀。此时有很多同学肯定会感吧，苦逼的IE6-8不支持，不敢使用。

**渐变**gradient

CSS3 Gradient 分为 [linear-gradient](http://lists.w3.org/Archives/Public/www-style/2011Apr/0083.html)（线性渐变）和 [radial-gradient](http://lists.w3.org/Archives/Public/www-style/2011Feb/0537.html)（径 向渐变）。而我们今天主要是针对线性渐变来剖析其具体的用法。为了更好的应用 CSS3 Gradient，我们需要先了解一下目前的几种现代浏览器的内核，主要有 Mozilla（Firefox，Flock等）、 WebKit（Safari、Chrome等）、Opera（Opera浏览器）、Trident（讨厌的IE浏览器）。

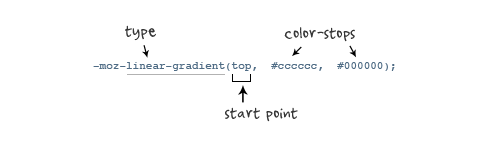
　　本文照常忽略IE不管，我们主要看看在 Mozilla、Webkit、Opera 下的应用，当然在 IE 下也可以实现，他需要通过 IE 特有的滤镜来实现，在后面会列出滤镜的使用语法，但不会具体介绍如何实用，感兴趣的可以搜索相关技术文档。

**一、线性渐变在 Mozilla 下的应用**

**语法：**

|  |  |
| --- | --- |
| 1 | -moz-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]\* ) |

**参数：**其共有三个参数，第一个参数表示线性渐变的方向，top 是从上到下、left 是从左到右，如果定义成 left top，那就是从左上角到右下角。第二个和第三个参数分别是起点颜色和终点颜色。你还可以在它们之间插入更多的参数，表示多种颜色的渐变。如图所示：



　　根据上面的介绍，我们先来看一个简单的例子：

**HTML：**

|  |  |
| --- | --- |
| 1 | <div class="example example1"></div> |

**CSS：**

|  |  |
| --- | --- |
| 1  2  3  4 | .example {     width: 150px;     height: 80px;   } |

　　如无特殊说明，我们后面的示例都是应用这一段 html 和 css 的基本代码。

　　现在我们给这个div应用一个简单的渐变样式：

|  |  |
| --- | --- |
| 1  2  3 | .example1{     background: -moz-linear-gradient( top,#ccc,#000);  } |

　　效果如下：

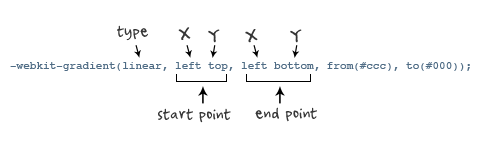


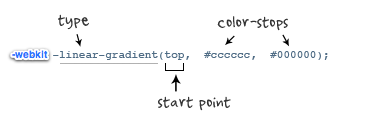
**二、线性渐变在 Webkit 下的应用**

**语法：**

|  |  |
| --- | --- |
| 1  2 | -webkit-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]\* )//最新发布书写语法  -webkit-gradient(<type>, <point> [, <radius>]?, <point> [, <radius>]? [, <stop>]\*) //老式语法书写规则 |

**参数：**-webkit-gradient 是 webkit 引擎对渐变的实现参数，一共有五个。第一个参数表示渐变类型（type），可以是linear（线性渐变）或者radial（径向渐变）。第二个参数和第 三个参数，都是一对值，分别表示渐变起点和终点。这对值可以用坐标形式表示，也可以用关键值表示，比如 left top（左上角）和left bottom（左下角）。第四个和第五个参数，分别是两个color-stop函数。color-stop 函数接受两个参数，第一个表示渐变的位置，0为起点，0.5为中点，1为结束点；第二个表示该点的颜色。如图所示：





　　我们先来看一个老式的写法示例：

|  |  |
| --- | --- |
| 1 | background: -webkit-gradient(linear,centertop,centerbottom,from(#ccc), to(#000)); |

　　效果如下所示：



　　接着我们在来看一下新式的写法：

|  |  |
| --- | --- |
| 1 | -webkit-linear-gradient(top,#ccc,#000); |

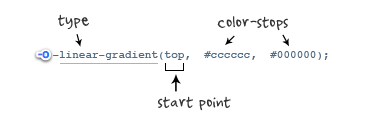
　　这个效果我就不在贴出来了，大家在浏览器中一看就明白了，他们是否一致的效果。仔细对比，在 Mozilla 和 Webkit 下两者的学法都基本上一致了，只是其前缀的区别，当然哪一天他们能统一成一样，对我们来说当然是更好了，那就不用去处理了。将大大节省我们的开发时间哟。

**三、线性渐变在 Opera 下的应用**

**语法：**

|  |  |
| --- | --- |
| 1 | -o-linear-gradient([<point> || <angle>,]? <stop>, <stop> [, <stop>]); /\* Opera 11.10+ \*/ |

**参数：**-o-linear-gradient 有三个参数。第一个参数表示线性渐变的方向，top 是从上到下、left 是从左到右，如果定义成 left top，那就是从左上角到右下角。第二个和第三个参数分别是起点颜色和终点颜色。你还可以在它们之间插入更多的参数，表示多种颜色的渐变。 （注：Opera 支持的版本有限，本例测试都是在 Opera11.1 版本下，后面不在提示），如图所示：



　　示例代码：

|  |  |
| --- | --- |
| 1 | background: -o-linear-gradient(top,#ccc, #000); |

　　效果如图所示：



**四、线性渐变在 Trident (IE) 下的应用**

**语法：**

|  |  |
| --- | --- |
| 1  2 | filter: progid:DXImageTransform.Microsoft.gradient(GradientType=0, startColorstr=#1471da, endColorstr=#1C85FB);/\*IE<9>\*/  -ms-filter: "progid:DXImageTransform.Microsoft.gradient (GradientType=0, startColorstr=#1471da, endColorstr=#1C85FB)";/\*IE8+\*/ |

　　IE依靠滤镜实现渐变。startColorstr表示起点的颜色，endColorstr 表示终点颜色。GradientType 表示渐变类型，0 为缺省值，表示垂直渐变，1 表示水平渐变。如图所示：



　　上面我们主要介绍了线性渐变在上述四大核心模块下的实现方法，接着我们主要针对线性渐变在 Mozilla、Webkit、Opera 三大模块下实现各种不同线性渐变实例：

　　从上面的语法中我们可以很清楚的知道，要创建一个线性渐变，我们需要创建一个起点和一个渐变方向（或角度），定义一个起始色：

|  |  |
| --- | --- |
| 1  2  3 | -moz-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]\* )  -webkit-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]\* )  -o-linear-gradient( [<point> || <angle>,]? <stop>, <stop> [, <stop>]\* ) |

　　具体应用如下：

|  |  |
| --- | --- |
| 1  2  3  4 | background:-moz-linear-gradient(left,#ace,#f96);/\*Mozilla\*/  background:-webkit-gradient(linear,050%,100%50%,from(#ace),to(#f96));/\*Old gradient for webkit\*/  background:-webkit-linear-gradient(left,#ace,#f96);/\*new gradient for Webkit\*/  background:-o-linear-gradient(left,#ace,#f96); /\*Opera11\*/ |

　　效果如下：



　　起始点（Starting Point）的工作方式类似于 background position。您可以设置水平和垂直位置为百分比，或以像素为单位，或在水平方向上可以使用left/center/right，在垂直方向上可以使 用top/center/bottom。位置起始于左上角。如果你不指定水平或垂直位置，它将默认为center。其工作方式主要包含：Top → Bottom、Left → Right、bottom → top、right → left等，接着我们主要一种一种来看其实现的效果：

**1、开始于center（水平方向）和top（垂直方向）也就是Top → Bottom：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* Firefox 3.6+ \*/  background: -moz-linear-gradient(top, #ace, #f96);  /\* Safari 4-5, Chrome 1-9 \*/  /\* -webkit-gradient(,  [, ]?,  [, ]? [, ]\*) \*/  background: -webkit-gradient(linear,top,from(#ace),to(#f96));  /\* Safari 5.1+, Chrome 10+ \*/  background: -webkit-linear-gradient(top, #ace, #f96);  /\* Opera 11.10+ \*/  background: -o-linear-gradient(top, #ace, #f96); |

　　效果：



**2、始于left（水平方向）和center（垂直方向）也是就Left → Right：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | /\* Firefox 3.6+ \*/  background: -moz-linear-gradient(left, #ace, #f96);  /\* Safari 5.1+, Chrome 10+ \*/  background: -webkit-linear-gradient(left, #ace, #f96);  /\* Opera 11.10+ \*/  background: -o-linear-gradient(left, #ace, #f96); |

　　效果如下：



**3、起始于left（水平方向）和top（垂直方向）:**

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(lefttop, #ace, #f96);  background: -webkit-linear-gradient(lefttop, #ace, #f96);  background: -o-linear-gradient(lefttop, #ace, #f96); |

　　效果如下：



**4、Linear Gradient (with Even Stops):**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | /\* Firefox 3.6+ \*/  background: -moz-linear-gradient(left, #ace, #f96, #ace, #f96, #ace);  /\* Safari 4-5, Chrome 1-9 \*/  background: -webkit-gradient(linear, lefttop, righttop, from(#ace), color-stop(0.25, #f96), color-stop(0.5, #ace), color-stop(0.75, #f96), to(#ace));  /\* Safari 5.1+, Chrome 10+ \*/  background: -webkit-linear-gradient(left, #ace, #f96, #ace, #f96, #ace);  /\* Opera 11.10+ \*/  background: -o-linear-gradient(left, #ace, #f96, #ace, #f96, #ace); |

　　效果如下：



**5、with Specified Arbitrary Stops：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | /\* Firefox 3.6+ \*/   background: -moz-linear-gradient(left, #ace, #f965%, #ace, #f9695%, #ace);   /\* Safari 4-5, Chrome 1-9 \*/   background: -webkit-gradient(linear, lefttop, righttop, from(#ace), color-stop(0.05, #f96), color-stop(0.5, #ace), color-stop(0.95, #f96), to(#ace));   /\* Safari 5.1+, Chrome 10+ \*/   background: -webkit-linear-gradient(left, #ace, #f965%, #ace, #f9695%, #ace);   /\* Opera 11.10+ \*/   background: -o-linear-gradient(left, #ace, #f965%, #ace, #f9695%, #ace); |

　　效果如下：



**6、角度(Angle)：**

　　正如上面看到的示例，如果您不指定一个角度，它会根据起始位置自动定义。如果你想更多的控制渐变的方向，您不妨设置角度试试。例如，下面的两个渐变具有相同的起点left center，但是加上一个30度的角度。

　　没有角度的示例代码：

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(left, #ace, #f96);  background: -webkit-linear-gradient(left,#ace,#f96);  background: -o-linear-gradient(left, #ace, #f96); |

　　加上30度的角度代码：

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(left30deg, #ace, #f96);  background: -webkit-gradient(linear, 00, 100%100%, from(#ace),to(#f96));  background: -o-linear-gradient(30deg, #ace, #f96); |

　　效果图如下：

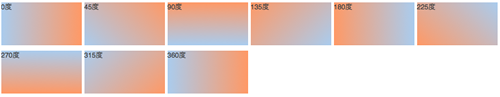
　　当指定的角度，请记住，它是一个由水平线与渐变线产生的的角度，逆时针方向。因此，使用0deg将产生一个左到右横向梯度，而90度将创建一个从底部到顶部的垂直渐变。我来看看你核心代码：

|  |  |
| --- | --- |
| 1  2  3  4 | background: -moz-linear-gradient(<angle>, #ace, #f96);  background: -webkit-gradient(<type>,<angle>, from(#ace), to(#f96));  background: -webkit-linear-gradient(<angle>, #ace, #f96);  background: -o-linear-gradient(<angle>, #ace, #f96); |

　　我们来看看各角度的区别：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | .deg0{    background: -moz-linear-gradient(0deg, #ace, #f96);    background: -webkit-gradient(linear,050%,100%50%,from(#ace),to(#f96));    background: -webkit-linear-gradient(0deg, #ace, #f96);    background: -o-linear-gradient(0deg, #ace, #f96);  }    .deg45{    background: -moz-linear-gradient(45deg, #ace, #f96);    background: -webkit-gradient(linear,0100%,100%0%,from(#ace),to(#f96));    background: -webkit-linear-gradient(45deg, #ace, #f96);    background: -o-linear-gradient(45deg, #ace, #f96);  }  .deg90{    background: -moz-linear-gradient(90deg, #ace, #f96);    background: -webkit-gradient(linear,50%100%,50%0%,from(#ace),to(#f96));    background: -webkit-linear-gradient(90deg, #ace, #f96);    background: -o-linear-gradient(90deg, #ace, #f96);  }  .deg135{    background: -moz-linear-gradient(135deg, #ace, #f96);    background: -webkit-gradient(linear,100%100%,00,from(#ace),to(#f96));    background: -webkit-linear-gradient(135deg, #ace, #f96);    background: -o-linear-gradient(135deg, #ace, #f96);  }  .deg180{    background: -moz-linear-gradient(180deg, #ace, #f96);    background: -webkit-gradient(linear,100%50%,050%,from(#ace),to(#f96));    background: -webkit-linear-gradient(180deg, #ace, #f96);    background: -o-linear-gradient(180deg, #ace, #f96);  }  .deg225{    background: -moz-linear-gradient(225deg, #ace, #f96);    background: -webkit-gradient(linear,100%0%,0100%,from(#ace),to(#f96));    background: -webkit-linear-gradient(225deg, #ace, #f96);    background: -o-linear-gradient(225deg, #ace, #f96);  }  .deg270{    background: -moz-linear-gradient(270deg, #ace, #f96);    background: -webkit-gradient(linear,50%0%,50%100%,from(#ace),to(#f96));    background: -webkit-linear-gradient(270deg, #ace, #f96);    background: -o-linear-gradient(270deg, #ace, #f96);  }  .deg315{    background: -moz-linear-gradient(315deg, #ace, #f96);    background: -webkit-gradient(linear,0%0%,100%100%,from(#ace),to(#f96));    background: -webkit-linear-gradient(315deg, #ace, #f96);    background: -o-linear-gradient(315deg, #ace, #f96);  }  .deg360{    background: -moz-linear-gradient(360deg, #ace, #f96);    background: -webkit-gradient(linear,050%,100%50%,from(#ace),to(#f96));    background: -webkit-linear-gradient(360deg, #ace, #f96);    background: -o-linear-gradient(360deg, #ace, #f96);  } |

　　效果如下：



　　除了起始位置和角度，你应该指定起止颜色。起止颜色是沿着渐变线，将会在指定位置（以百分比或长度设定）含有指定颜色的点。色彩的起止数是无限 的。如果您使用一个百分比位置，0％代表起点和100％是终点，但区域外的值可以被用来达到预期的效果。 这也是通过CSS3 Gradient制作渐变的一个关键所在，其直接影响了你的设计效果，像我们这里的示例都不是完美的效果，只是为了能给大家展示一个渐变的效果，大家就这 样先用着吧。我们接着看一下不同的起址色的示例：

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(top, #ace, #f9680%, #f96);  background: -webkit-linear-gradient(top,#ace,#f9680%,#f96);  background: -o-linear-gradient(top, #ace, #f9680%, #f96); |

　　效果如下：



　　如果没有指定位置，颜色会均匀分布。如下面的示例：

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(left, red, #f96, yellow, green, #ace);  background: -webkit-linear-gradient(left,red,#f96,yellow,green,#ace);  background: -o-linear-gradient(left, red, #f96, yellow, green, #ace); |

　　效果如下



**7、渐变上应用透明度(Transparency)：**

　　透明渐变对于制作一些特殊的效果是相当有用的，例如，当堆叠多个背景时。这里是两个背景的结合：一张图片，一个白色到透明的线性渐变。我们来看一个官网的示例吧：

|  |  |
| --- | --- |
| 1  2  3 | background: -moz-linear-gradient(right, rgba(255,255,255,0), rgba(255,255,255,1)),url(<http://demos.hacks.mozilla.org/openweb/resources/images/patterns/flowers-pattern.jpg>);  background: -webkit-linear-gradient(right, rgba(255,255,255,0), rgba(255,255,255,1)),url(<http://demos.hacks.mozilla.org/openweb/resources/images/patterns/flowers-pattern.jpg>);  background: -o-linear-gradient(right, rgba(255,255,255,0), rgba(255,255,255,1)),url(<http://demos.hacks.mozilla.org/openweb/resources/images/patterns/flowers-pattern.jpg>); |

　　接着看看效果吧



相关链接：

<http://www.cnblogs.com/lhb25/archive/2013/01/30/css3-linear-gradient.html>

**响应式布局**

随着3G的普及，越来越多的人使用手机上网。

移动设备正超过桌面设备，成为访问互联网的最常见终端。于是，网页设计师不得不面对一个难题：如何才能在不同大小的设备上呈现同样的网页？



手机的屏幕比较小，宽度通常在600像素以下；PC的屏幕宽度，一般都在1000像素以上（目前主流宽度是1366×768），有的还达到了2000像素。同样的内容，要在大小迥异的屏幕上，都呈现出满意的效果，并不是一件容易的事。

很多网站的解决方法，是为不同的设备提供不同的网页，比如专门提供一个mobile版本，或者iPhone / iPad版本。这样做固然保证了效果，但是比较麻烦，同时要维护好几个版本，而且如果一个网站有多个portal（入口），会大大增加架构设计的复杂度。

于是，很早就有人设想，能不能"一次设计，普遍适用"，让同一张网页自动适应不同大小的屏幕，根据屏幕宽度，自动调整布局（layout）？



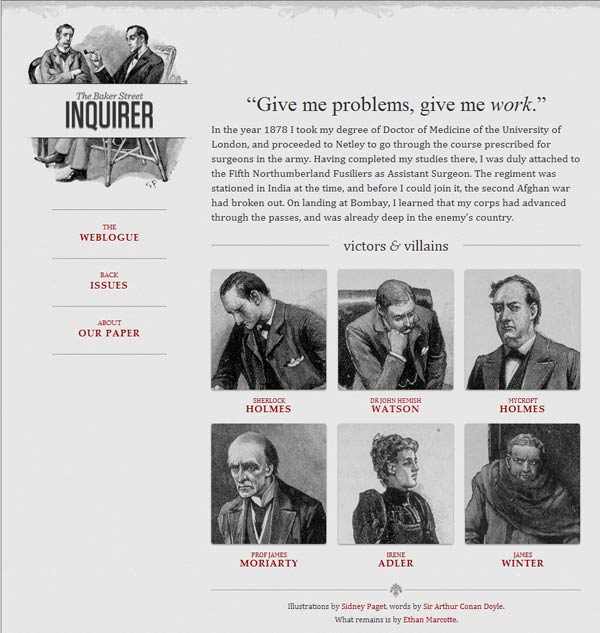
**一、"自适应网页设计"的概念**

2010年，Ethan Marcotte提出了["自适应网页设计"](http://www.alistapart.com/articles/responsive-web-design/)（Responsive Web Design）这个名词，指可以自动识别屏幕宽度、并做出相应调整的网页设计。

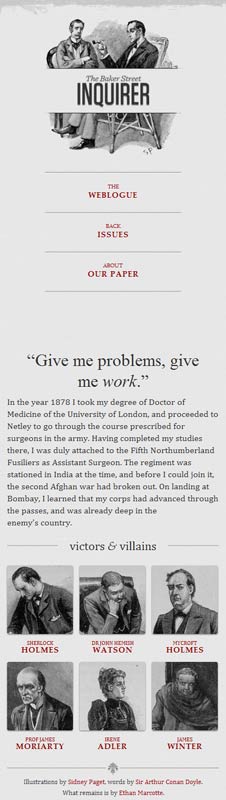
他制作了一个[范例](http://www.alistapart.com/d/responsive-web-design/ex/ex-site-flexible.html)，里面是《福尔摩斯历险记》六个主人公的头像。如果屏幕宽度大于1300像素，则6张图片并排在一行。



如果屏幕宽度在600像素到1300像素之间，则6张图片分成两行。



如果屏幕宽度在400像素到600像素之间，则导航栏移到网页头部。



如果屏幕宽度在400像素以下，则6张图片分成三行。



[mediaqueri.es](http://mediaqueri.es/)上面有更多这样的例子。

这里还有一个[测试小工具](http://www.benjaminkeen.com/misc/bricss/)，可以在一张网页上，同时显示不同分辨率屏幕的测试效果，我推荐安装。

**二、允许网页宽度自动调整**

"自适应网页设计"到底是怎么做到的？其实并不难。

首先，在网页代码的头部，加入一行[viewport元标签](https://developer.mozilla.org/en/mobile/viewport_meta_tag)。

　　<meta name="viewport" content="width=device-width, initial-scale=1" />

[viewport](https://developer.apple.com/library/ios/#DOCUMENTATION/AppleApplications/Reference/SafariWebContent/UsingtheViewport/UsingtheViewport.html)是网页默认的宽度和高度，上面这行代码的意思是，网页宽度默认等于屏幕宽度（width=device-width），原始缩放比例（initial-scale=1）为1.0，即网页初始大小占屏幕面积的100%。

所有主流浏览器都支持这个设置，包括IE9。对于那些老式浏览器（主要是IE6、7、8），需要使用[css3-mediaqueries.js](http://code.google.com/p/css3-mediaqueries-js/)。

　　<!--[if lt IE 9]>  
　　　　<script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>  
　　<![endif]-->

**三、不使用绝对宽度**

由于网页会根据屏幕宽度调整布局，所以不能使用绝对宽度的布局，也不能使用具有绝对宽度的元素。这一条非常重要。

具体说，CSS代码不能指定像素宽度：

　　width:xxx px;

只能指定百分比宽度：

　　width: xx%;

或者

　　width:auto;

**四、相对大小的字体**

字体也不能使用绝对大小（px），而只能使用相对大小（em）。

　　body {  
　　　　font: normal 100% Helvetica, Arial, sans-serif;  
　　}

上面的代码指定，字体大小是页面默认大小的100%，即16像素。

　　h1 {  
　　　　font-size: 1.5em;   
　　}

然后，h1的大小是默认大小的1.5倍，即24像素（24/16=1.5）。

　　small {  
　　　　font-size: 0.875em;  
　　}

small元素的大小是默认大小的0.875倍，即14像素（14/16=0.875）。

**五、流动布局（fluid grid）**

["流动布局"](http://www.alistapart.com/articles/fluidgrids/)的含义是，各个区块的位置都是浮动的，不是固定不变的。

　　.main {  
　　　　float: right;  
　　　　width: 70%;   
　　}

　　.leftBar {  
　　　　float: left;  
　　　　width: 25%;  
　　}

[float](http://designshack.net/articles/css/everything-you-never-knew-about-css-floats/)的好处是，如果宽度太小，放不下两个元素，后面的元素会自动滚动到前面元素的下方，不会在水平方向overflow（溢出），避免了水平滚动条的出现。

另外，绝对定位（position: absolute）的使用，也要非常小心。

**六、选择加载CSS**

"自适应网页设计"的核心，就是CSS3引入的[Media Query](http://www.w3.org/TR/CSS21/media.html)模块。

它的意思就是，自动探测屏幕宽度，然后加载相应的CSS文件。

　　<link rel="stylesheet" type="text/css"  
　　　　media="screen and (max-device-width: 400px)"  
　　　　href="tinyScreen.css" />

上面的代码意思是，如果屏幕宽度小于400像素（max-device-width: 400px），就加载tinyScreen.css文件。

　　<link rel="stylesheet" type="text/css"  
　　　　media="screen and (min-width: 400px) and (max-device-width: 600px)"  
　　　　href="smallScreen.css" />

如果屏幕宽度在400像素到600像素之间，则加载smallScreen.css文件。

除了用html标签加载CSS文件，还可以在现有CSS文件中加载。

　　@import url("tinyScreen.css") screen and (max-device-width: 400px);

**七、CSS的@media规则**

同一个CSS文件中，也可以根据不同的屏幕分辨率，选择应用不同的CSS规则。

　　@media screen and (max-device-width: 400px) {

　　　　.column {  
　　　　　　float: none;  
　　　　　　width:auto;  
　　　　}

　　　　#sidebar {  
　　　　　　display:none;  
　　　　}

　　}

上面的代码意思是，如果屏幕宽度小于400像素，则column块取消浮动（float:none）、宽度自动调节（width:auto），sidebar块不显示（display:none）。

**八、图片的自适应（fluid image）**

除了布局和文本，"自适应网页设计"还必须实现图片的[自动缩放](http://unstoppablerobotninja.com/entry/fluid-images)。

这只要一行CSS代码：

　　img { max-width: 100%;}

这行代码对于大多数嵌入网页的视频也有效，所以可以写成：

　　img, object { max-width: 100%;}

老版本的IE不支持max-width，所以只好写成：

　　img { width: 100%; }

此外，windows平台缩放图片时，可能出现图像失真现象。这时，可以尝试使用IE的[专有命令](http://css-tricks.com/ie-fix-bicubic-scaling-for-images/)：

　　img { -ms-interpolation-mode: bicubic; }

或者，Ethan Marcotte的[imgSizer.js](http://unstoppablerobotninja.com/demos/resize/imgSizer.js)。

　　addLoadEvent(function() {

　　　　var imgs = document.getElementById("content").getElementsByTagName("img");

　　　　imgSizer.collate(imgs);

　　});

不过，有条件的话，最好还是根据不同大小的屏幕，加载不同分辨率的图片。有[很多方法](http://blog.cloudfour.com/responsive-imgs-part-2/)可以做到这一条，服务器端和客户端都可以实现。